# How to Easily Add Comments to Your SAS Code

Authored by

**stats writer**

November 20, 2025

Effective use of comments is fundamental to robust and maintainable software development, and the SAS programming environment offers several powerful mechanisms for inserting descriptive notes into your code. Comments serve as vital code documentation, explaining the logic, intent, and functionality of specific sections of a program. They transform complex scripts into readable narratives, which is especially critical when collaborating with colleagues or revisiting older projects.

In SAS, there are three primary methods for creating comments: the traditional statement comment using an asterisk, the versatile C-style block comment, and the modern double-slash line comment. Mastering these techniques is an essential part of effective programming practice, ensuring that the purpose behind every data step, procedure, or macro remains transparent. This guide will explore each method in detail, providing practical examples to enhance the clarity and professionalism of your SAS scripts.

While manual entry is always an option, the SAS interface provides highly efficient shortcuts for toggling comments. The most user-friendly approach in environments like SAS Studio or the SAS Display Manager involves using a quick keyboard command, which streamlines the process of documenting or temporarily disabling lines of code. Understanding both manual and automated methods allows programmers to choose the technique best suited for the immediate task, whether it is documenting a single variable assignment or isolating a large block of code for debugging purposes.

## Utilizing the Traditional SAS Statement Comment

The oldest and arguably most distinctive method for adding a comment in SAS utilizes the asterisk symbol (*****) followed by the comment text, and it must be terminated by a semicolon (**;**). This method is native to SAS syntax and is particularly well-suited for adding short, informational notes that relate directly to a single statement or block of code. Unlike other comment types, the statement comment is processed as a complete SAS statement itself, meaning it is terminated by the standard semicolon.

When employing this method, it is crucial to remember the terminal semicolon. If the semicolon is omitted, SAS will attempt to interpret the subsequent line of code as part of the comment, potentially leading to compilation errors or unintended code behavior. Due to this strict requirement, the asterisk comment is generally used for single-line declarations, such as documenting the creation of a dataset or explaining a complex calculation within a DATA step.

While effective for concise declarations, the statement comment is less ideal for multi-line documentation headers or temporarily disabling large blocks of code, as it requires the ***comment;** structure to be repeated on every line. For instance, if you were documenting a variable list, you would write: *** Variable list for input file;** or *** Calculate mean height;**. This distinction is

important when deciding which comment type provides the best balance of readability and efficiency for a given programming task.

## Implementing the Versatile Block Comment (/* */)

The second primary method, and perhaps the most flexible, is the C-style block comment, denoted by starting the comment with **/*** and ending it with ***/**. This structure allows the programmer to span documentation across multiple lines without needing repetitive delimiters. It is the preferred method for creating extensive documentation headers at the beginning of a program or for temporarily disabling large sections of SAS code during debugging.

A significant advantage of the block comment is its ability to enclose virtually any amount of text, making it perfect for detailed explanations of algorithm choices, system dependencies, or revision history. Furthermore, unlike the traditional asterisk comment, the block comment does not require a semicolon for termination, treating everything between the start and end markers as pure, non-executable text.

However, a critical limitation to note when using the **/* */** structure in older or certain versions of SAS is the restriction on nesting. You cannot typically place one block comment inside another block comment. If you attempt to comment out a section of code that already contains internal block comments, the outer comment will terminate prematurely at the first instance of ***/**, leading to unexpected compilation errors for the remaining uncommented code. Careful use and awareness of this limitation are essential for maintaining clean and error-free scripts.

## Adopting the Modern Line Comment (//)

The third form of commenting available in modern SAS environments, such as SAS 9.4 and SAS Studio, is the double-slash line comment (**//**). Inherited from C++ and Java syntax, this technique is used to comment out the remainder of a line following the double slashes. It provides a quick and clean way to add inline commentary or to quickly disable a single line of code without needing a corresponding closing delimiter.

The primary benefit of the double-slash method is its simplicity and visual distinction. It is excellent for quickly annotating a specific variable assignment or parameter within a procedure call. For example, you might see code structured like: `PROC PRINT DATA=mylib.data; // Print initial dataset`. This method is highly favored by programmers coming from other contemporary programming languages due to its immediacy and ease of use.

It is important for developers to check the compatibility of the **//** syntax with their specific SAS environment. While widely supported in newer versions and interfaces, older installations might not recognize this structure, requiring reliance on the traditional asterisk or the block comment syntax.

For cross-platform or legacy code bases, utilizing the asterisk or block comment ensures maximum compatibility, solidifying the importance of understanding all three options.

## Leveraging Keyboard Shortcuts for Commenting Efficiency

While manual insertion of comment delimiters (***/* */*** or **** ;***) is possible, the most efficient method for toggling comments in the SAS Editor or SAS Studio is through the use of a simple keyboard shortcut. This method significantly speeds up the process of code documentation and is particularly useful during the debugging phase when large sections of code need to be quickly enabled or disabled for testing.

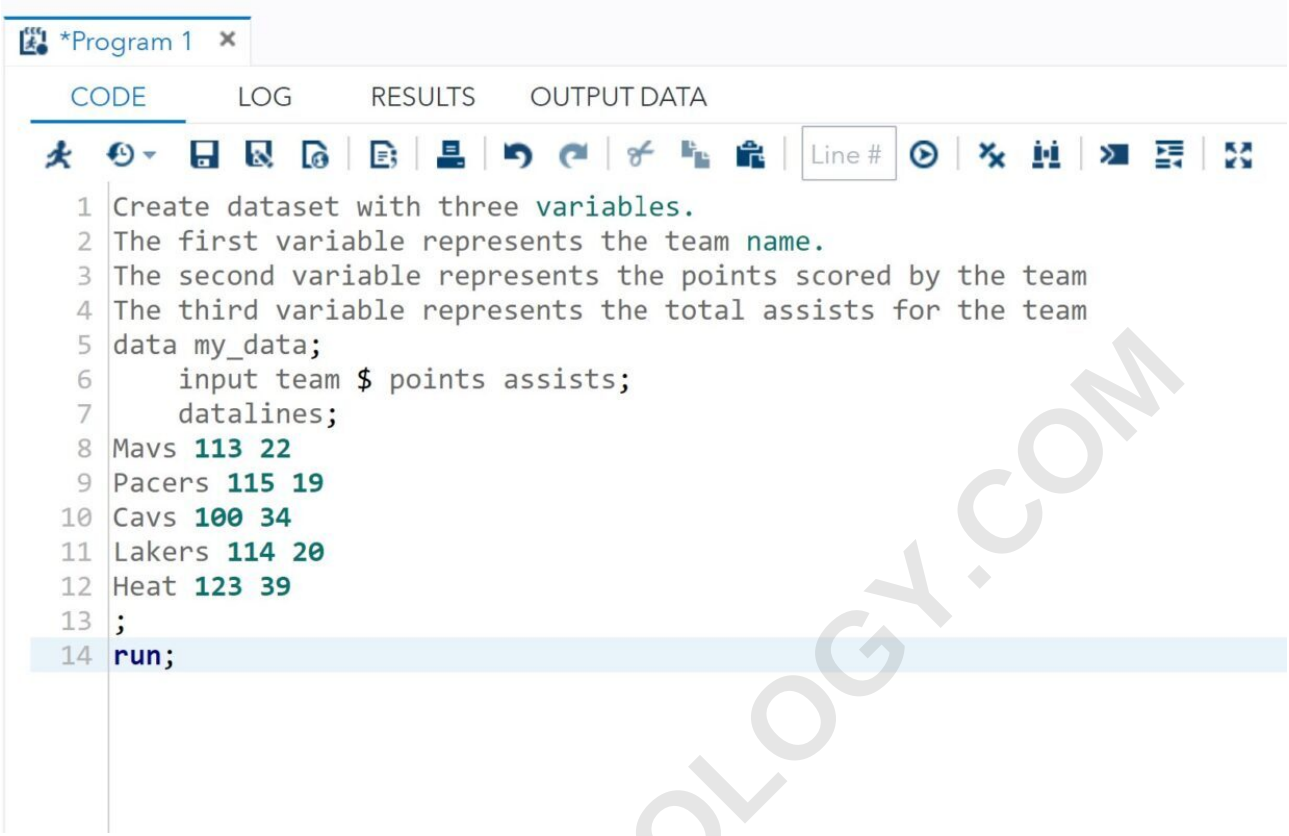
The easiest and fastest way to wrap selected text within block comments is to highlight the desired lines and press the combination of keys: **Ctrl** + ***/***. This action automatically inserts the opening delimiter (***/****) and the closing delimiter (****/***) around the highlighted block. If the text already contains a comment, pressing the shortcut again acts as an uncomment feature, intelligently removing the surrounding delimiters and restoring the code to its executable state.

This powerful toggle functionality means that the same shortcut is used for both commenting and uncommenting. The system detects whether the selected block is currently commented out and executes the corresponding action. This symmetrical design minimizes cognitive load and allows the programmer to focus on the structure and logic of the code rather than the manual insertion of SAS syntax elements.

## Detailed Example: Applying Block Comments via Shortcut

To illustrate the practical application of the keyboard shortcut method, consider a scenario where you have an existing section of SAS code that requires a descriptive header or needs its initial setup lines temporarily neutralized. Effective documentation starts with clearly separating functional code from explanatory notes.

Suppose we begin with the following raw SAS code, where the initial lines are intended to describe the data manipulation that follows. These lines, however, are currently active code and would generate errors if not properly converted into comments:



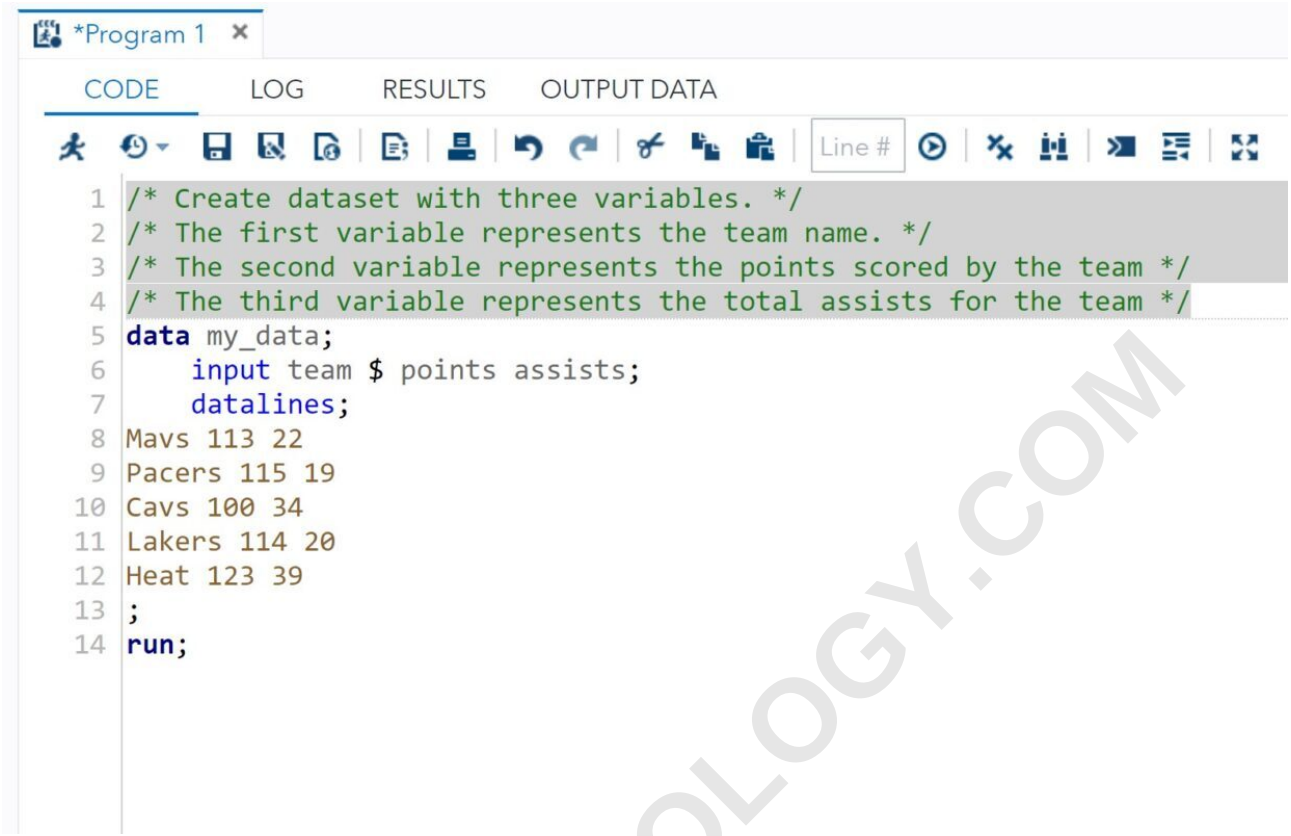



The primary objective is to convert the first four lines of text, which summarize the program's intent, into non-executable comments. To achieve this efficiently, we must use the automated feature provided by the SAS editor.

To transform these lines into valid comments, we simply highlight all four rows using the cursor. Once the selection is made, execute the keyboard command **Ctrl** + **/**. The editor instantly applies the block comment syntax to enclose the entire selection, successfully turning the descriptive text into inert documentation.

## Observing the Outcome of the Commenting Action

After executing the **Ctrl** + **/** command on the selected lines, the SAS Editor immediately displays the result, confirming that the code has been commented out. Observe the structure of the resulting code below:






Notice the strategic placement of the delimiters: a forward slash followed by an asterisk (***/***) has been inserted automatically at the beginning of the first selected line, and an asterisk followed by a forward slash (***/***) has been added to the end of the last selected line. This visual cue clearly indicates that the encompassed section, despite spanning multiple lines, is now treated as a single block comment by the SAS compiler.

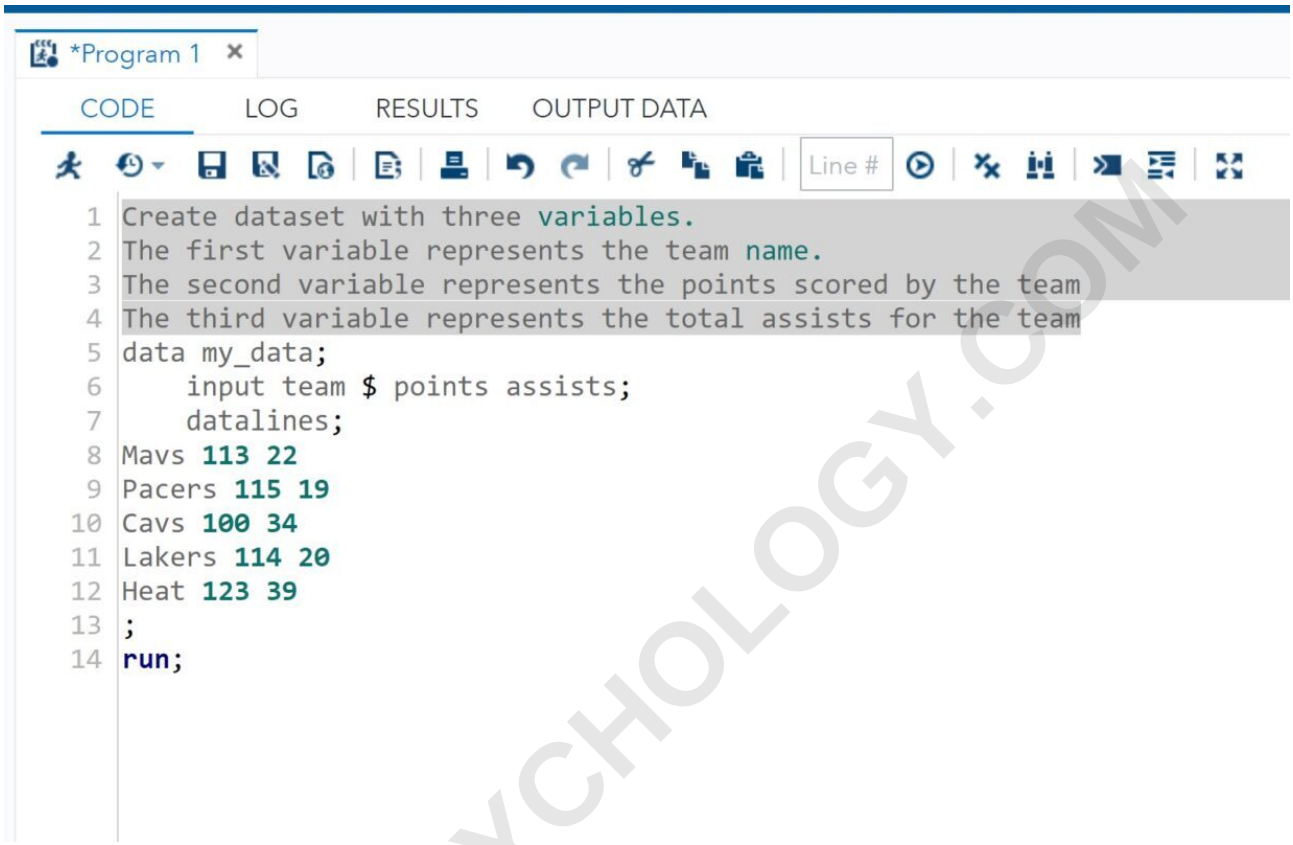
This block is now successfully excluded from execution when the program runs. The elegance of this solution lies in its ability to handle multi-line selections seamlessly, ensuring that the documentation remains intact and visually separated from the active logic of the program. This functionality is pivotal for maintaining high standards of code documentation and supporting complex development cycles.

## Reversing the Commenting Process (Uncommenting)

Just as easily as the comments were applied, the SAS editor allows the programmer to reverse the action. To restore the commented lines back into active, executable code, the process remains exactly the same: highlight the block that is currently enclosed by the ***/* */*** delimiters and press the toggle shortcut **Ctrl** + ***/*** once again.

This action instructs the editor to intelligently detect the presence of the block comment delimiters

surrounding the selection and remove them, resulting in the return of the original code state. This is an indispensable feature for debugging, allowing developers to swiftly isolate, test, and then reintegrate sections of code.






As demonstrated in the image above, the forward slashes and asterisks have been successfully stripped away from the first four lines. These lines are now considered active SAS syntax and will be executed upon program submission. The efficiency of the toggle command is a cornerstone of productivity within the SAS environment.

## Strategic Best Practices for SAS Commenting

Choosing the right type of comment for a specific task dramatically improves code readability and maintainability. While all three methods (*** ;**, **/* */**, and **//**) achieve the same goal--excluding text from execution--their structural differences make them suitable for distinct purposes in SAS programming practice. Adopting a standardized approach across a project team ensures consistency and reduces ambiguity.

Code documentation best practices suggest using block comments (**/* */**) for large, multi-line program headers that include metadata such as author, date, purpose, and version history. For instance, a program header should explicitly state the required input datasets and the intended

output datasets. Conversely, the statement comment (*** ;**) is perfect for documenting macro variables or explaining parameters within a procedure step, as its mandatory semicolon termination fits naturally into the flow of SAS statements.

The double-slash (***//***) should be reserved for quick, precise, inline notes that explain the immediate logic of a calculation or a conditional statement, especially when integrating with other languages or utilizing complex expressions where clarity is paramount. Furthermore, whenever possible, programmers should utilize the keyboard shortcuts to toggle block comments for rapid debugging, as this minimizes the risk of manual syntax errors that can occur when typing delimiters repeatedly.

Finally, remember that effective commenting goes beyond simply explaining **what** the code does; it should focus on explaining **why** the code is structured in a particular way, especially concerning non-obvious business logic or workarounds for known data limitations. Sparse or poorly written comments can be worse than no comments at all, as they may mislead future maintainers.

## Summary of Comment Types in SAS

To provide a quick reference for the appropriate use case for each SAS comment type, the following list summarizes the syntax and ideal applications discussed throughout this article:

**Statement Comment (`* text;`):**
Used for single-line notes, especially within the flow of existing SAS syntax statements. Must always end with a semicolon (**;**). Best for documenting variable assignments or single parameters.

**Block Comment (`/* text */`):**
Used for multi-line documentation, program headers, and temporarily disabling large sections of code. This is the only method that can reliably enclose text spanning many lines without requiring repetition. Cannot be nested.

**Line Comment (`// text`):**
Used in modern SAS versions (like SAS Studio) to comment out the rest of the line from the point of the double-slash onwards. Excellent for quick inline annotations and explanations.

By integrating these varied commenting techniques, SAS developers can significantly elevate the quality, maintainability, and longevity of their codebases, transforming complex analytical scripts into well-organized, accessible programming assets.

Mastery of these fundamental programming practice elements ensures compliance with high standards of professional coding, leading to more reliable and collaborative development efforts.

## Further Resources for SAS Operations

Understanding how to effectively manage comments is just one aspect of comprehensive SAS expertise. For those looking to expand their skills further within the platform, the following resources provide guidance on performing other common and essential data manipulation and analytical operations within the SAS environment.

These tutorials cover foundational skills necessary for data preparation and reporting, building upon the principles of clean code demonstrated through proper commenting.

The following tutorials explain how to perform other common operations in SAS: